AMENDMENTS TO THE CLAIMS

1. (currently amended) A method, comprising the steps of:

concurrently executing on a single computer a first operating system coded in a RISC instruction set, being an instruction set having a fixed-length instruction format and a load/store/operate organization, and a second operating system coded in a CISC instruction set, being an instruction set with variable-length instructions and many instructions having multiple side-effects, the CISC operating system being unmodified for execution on the computer of the RISC instruction set; and

accepting an exception raised while executing a program coded in the RISC instruction set, and in response, initiating ~~an~~ execution of a thread under the CISC operating system;

delivering the exception to the initiated thread for handling by the CISC operating system.

2. (original) A method, comprising the steps of:

concurrently executing on a single computer a first operating system coded in a RISC instruction set, being an instruction set having a fixed-length instruction format and a load/store/operate organization, and a second operating system coded in a CISC instruction set, being an instruction set with variable-length instructions and many instructions having multiple side-effects, the CISC operating system being unmodified for execution on the computer of the RISC instruction set; and

accepting an exception occurring during execution of a program coded in the RISC instruction set, and routing the exception for handling in the CISC operating system.

3. (original) The method of claim 2, further comprising the step of:

accepting an exception occurring during execution of a program coded in the CISC instruction set, and routing the exception for handling in the RISC operating system.

4. (original) The method of claim 3, wherein the RISC operating system comprises a collection of interrupt service routines programmed to emulate instructions in the CISC instruction set.

5. (original) The method of claim 3, wherein:

the computer has a file of general registers, the CISC instruction set providing accessibility to only a subset of the general register file, intermediate results of instructions of the CISC instruction set being stored in registers of the general register file that are inaccessible in the CISC instruction set.

6. (original) The method of claim 3, wherein:

during execution of an instruction on the computer, in response to an operation of the instruction calling for an architecturally-visible side-effect in an architecturally-visible storage location, storing a value representative of an architecturally-visible representation of the side-effect, a format of the representative value being different than an architecturally-visible representation of the side-effect, and resuming the execution without generating the architecturally-visible side-effect;

later writing the architecturally-visible representation corresponding to the representative value into the architecturally-visible storage location.

7. (original) The method of claim 3, further comprising the steps of:

recognizing a condition that is a superset of a condition being monitored for occurrence, and raising a first exception as a result of recognizing the superset condition;

in software, filtering the superset condition to determine whether the monitored condition has occurred;

if the monitored condition has occurred, establishing a second exception to be raised after execution of further instructions of the instruction stream.

8. (original) The method of claim 3, further comprising the step of:

recognizing a condition in which a first CISC instruction is to affect the execution of a second CISC instruction, and in response, setting the processor into single-step mode;

taking a single-step exception after executing the second CISC instruction, and setting the processor out of single-step mode.

9. (original) The method of claim 3, further comprising the steps of:

delaying accepting exception occurring at an intermediate point of execution of a CISC instruction until an CISC instruction boundary.

10. (original) The method of claim 3, further comprising the step of:

in an exception handler coded in the RISC instruction set, saving a portion of the context of the computer, and altering the context of the excepted program before delivering the exception to the CISC operating system.

11. (original) The method of claim 2, further comprising the step of::

without modifying the CISC operating system, establishing an entry handler for execution at a specified entry point or on a specified entry condition to the CISC operating system, the entry handler programmed to save a context of an excepted program and modify the program context before delivering the modified context to the CISC operating system.

12. (original) The method of claim 2, wherein:

the RISC instructions and CISC instructions are executed in a common execution pipeline.

13. (currently amended) The method of claim 2, wherein:

while executing a program coded in the RISC instruction set, receiving an exception;

in response to the exception, initiating ~~an~~ execution of a thread in the CISC operating system;

delivering the exception to the initiated thread for handling by the CISC operating system.

14. (original) The method of claim 2, wherein:

executing an instruction coded in the RISC instruction set, the instruction storing into a memory location a value of an instruction coded in the CISC instruction set;

in response to the storing, marking as stale copies of the memory location, including copies in any instruction cache;

executing the CISC instruction in the execution pipeline.

15. (original) The method of claim 2, further comprising the step of:

in the RISC operating system, building an exception frame on a memory stack before tending execution to the CISC operating system.

16. (original) The method of claim 2, wherein:

the exception is a synchronous fault generated by a RISC instruction.

17. (original) The method of claim 2, wherein:

the exception is a trap requesting a file access service from the CISC operating system on behalf of the program.

18. (original) A computer, comprising:

a first operating system loaded into memory, being coded in a RISC instruction set, being an instruction set having a fixed-length instruction format and a load/store/operate organization; and

a second operating system loaded into memory, being coded in a CISC instruction set, being an instruction set with variable-length instructions and many instructions having multiple side-effects, the CISC operating system being unmodified for execution on the computer of the RISC instruction set; and

hardware and/or software designed to accept an exception occurring during execution of a program coded in the RISC instruction set, and to route the exception for handling in the CISC operating system.

19. (original) The computer of claim 18, further comprising:

hardware and/or software designed to accept an exception occurring during execution of a program coded in the CISC instruction set, and to route the exception for handling in the RISC operating system.

20. (original) The computer of claim 19, further comprising:

an instruction decoder for the CISC instruction set designed to issue, for at least some of the decoded CISC instructions, two or more instructions in the RISC instruction set into the execution pipeline.

21. (original) The computer of claim 20, further comprising:

pipeline exception circuitry, effective on recognizing an exception occurring in a program coded in the CISC instruction set, to architecturally expose in processor registers of the computer information describing a processor state of the computer, and to transfer execution to an exception handler; and

pipeline resumption circuitry effective after completion of the software exception handler to resume execution of the excepted program based on the information in the processor registers;

the processor registers and general purpose registers of the computer architecturally exposing sufficient processor state and providing sufficient working storage for execution of the exception handler and resumption of the program, without storing processor state to the main memory.

22. (original) The computer of claim 20, further comprising:

a register and control logic for that register that capture and architecturally expose an intra-instruction program counter value when an instruction of the CISC instruction set raises an exception at an intermediate point.

23. (original) The computer of claim 19, wherein:

the RISC instructions and CISC instructions are executed in a common execution pipeline.

24. (original) The computer of claim 19, wherein:

at least a portion of the implementation of the CISC instruction set includes emulation routines coded in the RISC instruction set.

25. (original) The computer of claim 19, further comprising:

processor register control circuitry designed to store information describing the decoding of the CISC instructions into architecturally-visible processor registers of the computer.

26. (original) The computer of claim 19, further comprising:

pipeline control circuitry designed to recognize an exception occurring in a CISC instruction after a first side-effect of the CISC instruction has been architecturally committed, to transfer control to a software exception handler for the first exception, and to resume execution of the excepted CISC instruction after completion of the exception handler, processor registers of the computer being designed to capture sufficient information about the state of the excepted instruction that the transfer and resume are effected without saving intermediate results of the excepted CISC instruction on a memory stack.

27. (original) The computer of claim 19, further comprising:

a multi-stage execution pipeline; and

an instruction decoder designed to generate information descriptive of instructions to be executed by the pipeline, and to store the information into a non-pipelined register of the computer;

the instruction decoder being designed to determine whether instructions will complete in the pipeline, and to abstain from writing descriptive information into the register for instructions following an instruction determined not to complete.

28. (original) The computer of claim 19, wherein a class of exceptions is handled in part in each of the CISC and RISC operating systems.

29. (original) The computer of claim 18, further comprising:

store monitoring circuitry designed to monitor store instructions executed by the computer and to invalidate any copies of a datum in memory overwritten by the store instructions, including copies of instructions in any instruction cache in the instruction set other than the instruction set of the current store instruction.

30. (original) The computer of claim 18, further comprising a memory management unit designed to manage the storage of instructions of the RISC and CISC instruction sets between a main memory of the computer and one or more cache levels.

31. (currently amended) A method, comprising the steps of:

in response to an exception raised while executing a thread of a program coded in instructions of a first instruction set architecture, delivering the exception to an execution thread for execution of a handler for the exception, the handler's thread being distinct from the thread in which the program was executing, the handler's thread being ~~initiating~~ an execution ~~of a~~ thread under an operating system coded in instructions of a second instruction set architecture, the handler being a handler of the operating system;

~~delivering the exception to the initiated thread for handling by the operating system~~.

32. (original) The method of claim 31, further comprising:

executing a program in a computer comprising a hardware instruction decoder implementing less than an entire architectural definition of the first instruction set, a remainder of the first instruction set being implemented in a software emulator.

33. (original) The method of claim 31:

wherein the first instruction set is a complex instruction set having variable-length instructions and many instructions having multiple side-effects;

and further comprising the step of storing information describing the decoding of the complex instructions into architecturally-visible processor registers of the computer.

34. (original) The method of claim 31, further comprising the steps of:

during execution of an instruction on the computer, in response to an operation of the instruction calling for an architecturally-visible side-effect in an architecturally-visible storage location, storing a value representative of an architecturally-visible representation of the side-effect, a format of the representative value being different than an architecturally-visible representation of the side-effect, and resuming the execution without generating the architecturally-visible side-effect;

later writing the architecturally-visible representation corresponding to the representative value into the architecturally-visible storage location.

35. (original) The method of claim 31, further comprising the step of:

in an exception handler coded in the first instruction set, saving a portion of the context of the computer, and altering the context of the excepted program before delivering the exception to the operating system.

36. (previously presented) The method of claim 31:

wherein the second instruction set is a CISC instruction set, and the operating system coded in instructions of a second instruction set is a pre-existing, off-the-shelf operating system;

further comprising the step of without modifying the CISC operating system, establishing an entry handler for execution at a specified entry point or on a specified entry condition to the CISC operating system, the entry handler programmed to save a context of an excepted program and modify the program context before delivering the modified context to the CISC operating system.

37. (original) The method of claim 31, wherein:

the first instruction set and second instruction set are executed in a common execution pipeline.

38. (currently amended) A computer, comprising:

hardware and/or software designed to respond an exception raised while executing a thread of a program coded in instructions of a first instruction set architecture of the computer by delivering the exception to an execution thread for execution of a handler for the exception, the handler's thread being distinct from the thread in which the program was executing, the handler's thread being ~~initiating~~ an execution ~~of a~~ thread under an operating system coded in instructions of a second instruction set architecture and running on the computer, the handler being a handler of the operating system;

~~hardware and/or software designed to deliver the exception to the initiated thread for handling by the operating system~~.

39. (original) The computer of claim 38, further comprising:

pipeline exception circuitry, effective on recognizing an exception occurring in the program, to architecturally expose in processor registers of the computer information describing a processor state of the computer, and to transfer execution to an exception handler; and

pipeline resumption circuitry effective after completion of the software exception handler to resume execution of the program based on the information in the processor registers;

the processor registers and general purpose registers of the computer architecturally exposing sufficient processor state and providing sufficient working storage for execution of the exception handler and resumption of the program, without storing processor state to the main memory.

40. (original) The computer of claim 38, wherein:

the first instruction set is a RISC instruction set, being an instruction set having a fixed-length instruction format and a load/store/operate organization; and

the second instruction set is a CISC instruction set, being an instruction set with variable-length instructions and many instructions having multiple side-effects.

41. (previously presented) The computer of claim 40, wherein:

the RISC instructions and CISC instructions are executed in a common execution pipeline.

42. (previously presented) The computer of claim 40, further comprising:

pipeline control circuitry designed to recognize an exception occurring in a CISC instruction after a first side-effect of the CISC instruction has been architecturally committed, to transfer control to a software exception handler for the first exception, and to resume execution of the excepted CISC instruction after completion of the exception handler, processor registers of the computer being designed to capture sufficient information about the state of the excepted instruction that the transfer and resume are effected without saving intermediate results of the excepted CISC instruction on a memory stack.

43. (previously presented) The computer of claim 40, wherein a class of exceptions is handled in part in operating systems coded for each of the CISC and RISC instruction sets.

44. (new) The method of claim 31, further comprising the step of:

signaling the operating system to start up a new thread, being the handler's thread.